

EXHIBIT B

```
package KnowledgeAgents;

import se.*;
import gui.*;
5 import utils.*;

import java.awt.*;
import java.awt.event.*;
import java.util.Stack;
10 import java.io.*;

/**
 * The user interface of the Agent.
 */
15 public class AgentGUI extends Frame
    implements ActionListener, ItemListener
{
    Agent agent;
    Trace trace;
20
    /* private gui instance variables */
    CheckboxGroup    queryType;
    Checkbox        textQuery, linkQuery, repSites;
    private TextField tfQuery;
25
    CheckboxGroup    engineOptions;
    Checkbox        allEngines, selectEngines;
    List            engines;
30
    Checkbox        fileSelect;
    private TextField tfOutputFile;

    private TextField tfExpandFactor, tfRootSetSize, tfNumResults;
    private TextField tfMaxLinkWeight, tfQueryRefineFactor;
35
    private TextField tfNSites, tfPhase, tfLeft;
    private TextField tfServQIn, tfServQOut, tfServQUtil;
    private boolean   vis;
    private Button    queryRefine, addSites;
40    private Button    searchLearn,searchOnly;

    /* public methods */

    /**
45    public AgentGUI(Agent  agent,
                      Trace  trace )
    {
        this.agent = agent;
        this.trace = trace;
50
        int  gridy = 0, gridx = 1;
        Font fntTitle = new Font("Helvetica",  Font.ITALIC, 18);
        Font fnt      = new Font("TimesRoman",  Font.PLAIN, 12);
        Font fntBold  = new Font("TimesRoman",  Font.BOLD, 12);
55
        Panel      p      = new Panel();
        ScrollPane scroll  = new ScrollPane();
        GridBagLayout gbl   = new GridBagLayout();
        GridBagConstraints c     = new GridBagConstraints();
60
        vis = true;
    }
}
```

```

    p.setFont (fnt);
    p.setLayout(gbl);
65
    // prepare the panel's title
/*
    GuiAddFields.AddLabel ("Knowledge Agent Window",
                           true,
70
                           0, gridy++, 4, 1,
                           GridBagConstraints.CENTER,
                           fntTitle,
                           gbl, p );
    GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
75
    GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
*/
//***** *****
80
    GuiAddFields.AddLabel
        ("Enter Query/Add Repository Sites",
         false,
         0, gridy++, 2, 1,
         GridBagConstraints.CENTER,
         fntBold,
         gbl, p );
85

    // add the radio buttons and the file dialog
    queryType = new CheckboxGroup();
    textQuery = new Checkbox ("Text query", queryType, true);
90
    linkQuery = new Checkbox ("Sample URL query", queryType, false);
    repSites = new Checkbox ("Repository Sites", queryType, false);

    GuiAddFields.AddCheckbox(textQuery, 0, gridy, 1,1,gbl,p,(Frame)this);
    tfQuery = GuiAddFields.AddTextField
95
        (45,
         1, gridy++, 4, 1,
         GridBagConstraints.WEST,
         false,
         gbl, p );
100
    tfQuery.setEditable(true);
    GuiAddFields.AddCheckbox(linkQuery, 0, gridy, 1,1,gbl,p,(Frame)this);
    GuiAddFields.AddCheckbox(repSites, 0, gridy+1, 1,1,gbl,p,(Frame)this);

    GuiAddFields.AddLabel ("      Query Refinement Factor:",
105
                           false,
                           1, gridy, 2, 1,
                           GridBagConstraints.CENTER,
                           fnt,
                           gbl, p );
110
    tfQueryRefineFactor= GuiAddFields.AddTextField
        (5,
         3, gridy, 1, 1,
         GridBagConstraints.WEST,
         false,
         gbl, p );
115
    tfQueryRefineFactor.setText("2");
    queryRefine = GuiAddFields.AddButton ("Refine Query", 4, gridy++,
                                         GridBagConstraints.CENTER,
                                         gbl, p, (Frame)this);
120
    addSites     = GuiAddFields.AddButton ("Add Sites", 1, gridy++,
                                         GridBagConstraints.WEST,
                                         gbl, p, (Frame)this);

```

```

125 GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
//*****
130 GuiAddFields.AddLabel
        ("Choose Search Engines:",
         false,
         0, gridy, 2, 1,
         GridBagConstraints.CENTER,
         fntBold,
         gbl, p
        );
135 GuiAddFields.AddLabel
        ("Various Arguments:",
         false,
         3, gridy++, 2, 1,
         GridBagConstraints.CENTER,
         fntBold,
         gbl, p
        );
140 // add the radio buttons and the scrollable list
engineOptions = new CheckboxGroup();
allEngines    = new Checkbox ("All Engines", engineOptions, false);
selectEngines = new Checkbox ("Select From List --> ", .
                           engineOptions, true
                           );
145

150 GuiAddFields.AddCheckbox(allEngines,
                           0, gridy, 1,1,gbl,p,(Frame)this);
GuiAddFields.AddCheckbox(selectEngines,
                           0, gridy+1, 1,1,gbl,p,(Frame)this);

155 engines = new List(SearchEngine.NUM_ENGINES,true);
for ( int i=0; i<SearchEngine.NUM_ENGINES; i++ )
    engines.addItem(SearchEngine.engineNames[i]);
GuiAddFields.AddComponent (engines, 1, gridy,1,2,gbl,p);

160 GuiAddFields.AddLabel ("Max. Link Weight:",
                        false,
                        3, gridy, 2, 1,
                        GridBagConstraints.CENTER,
                        fnt,
                        gbl, p
                       );
165 tfMaxLinkWeight = GuiAddFields.AddTextField
                     (5,
                      5, gridy, 1, 1,
                      GridBagConstraints.WEST,
                      false,
                      gbl, p
                     );
170 tfMaxLinkWeight.setText("1.25");

175 gridy+=2;
GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
//*****
180 GuiAddFields.AddLabel
        ("Set Size Restrictions",
         false,
         0, gridy++, 2, 1,
         GridBagConstraints.CENTER,
         fntBold,
         gbl, p
        );

```

```

185 GuiAddFields.AddLabel ("Root Set Size:",
186           false,
187           0, gridy, 1, 1,
188           GridBagConstraints.CENTER,
189           fnt,
190           gbl, p );
190 tfRootSetSize = GuiAddFields.AddTextField
191   (3,
192     1, gridy, 1, 1,
193     GridBagConstraints.WEST,
194     false,
195     gbl, p );
195 tfRootSetSize.setText("30");

200 GuiAddFields.AddLabel ("Expand Factor:",
201           false,
202           2, gridy, 1, 1,
203           GridBagConstraints.CENTER,
204           fnt,
205           gbl, p );
205 tfExpandFactor = GuiAddFields.AddTextField
206   (3,
207     3, gridy, 1, 1,
208     GridBagConstraints.WEST,
209     false,
210     gbl, p );
210 tfExpandFactor.setText("40");

215 GuiAddFields.AddLabel ("Show Best:",
216           false,
217           4, gridy, 1, 1,
218           GridBagConstraints.CENTER,
219           fnt,
220           gbl, p );
220 tfNumResults = GuiAddFields.AddTextField
221   (3,
222     5, gridy++, 1, 1,
223     GridBagConstraints.WEST,
224     false,
225     gbl, p );
225 tfNumResults.setText("15");

226 GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);

//***** *****
230 GuiAddFields.AddLabel
231   ("Execution Trace",
232     false,
233     0, gridy++, 1, 1,
234     GridBagConstraints.CENTER,
235     fntBold,
236     gbl, p );
236

240 GuiAddFields.AddLabel ("Number Of Sites:",
241           false,
242           0, gridy, 1, 1,
243           GridBagConstraints.CENTER,
244           fnt,
245           gbl, p );
245 tfNSites = GuiAddFields.AddTextField

```

```

245                               (5,
246                               1, gridy, 1, 1,
247                               GridBagConstraints.WEST,
248                               false,
249                               gbl, p );
250   tfNSites.setText("0");
251   tfNSites.setEditable(false);

255   GuiAddFields.AddLabel ("URLs Left:",
256                           false,
257                           2, gridy, 1, 1,
258                           GridBagConstraints.CENTER,
259                           fnt,
260                           gbl, p );
261   tfLeft = GuiAddFields.AddTextField
262           (4,
263            3, gridy, 1, 1,
264            GridBagConstraints.CENTER,
265            false,
266            gbl, p );
267   tfLeft.setText("");
268   tfLeft.setEditable(false);

270   GuiAddFields.AddLabel
271       ("Dedicated Rdrs:",
272        false,
273        4, gridy, 1, 1,
274        GridBagConstraints.CENTER,
275        fnt,
276        gbl, p );
277   tfServQUtil= GuiAddFields.AddTextField
278       (3,
279        5, gridy++, 1, 1,
280        GridBagConstraints.WEST,
281        false,
282        gbl, p );
283   tfServQUtil.setText("");
284   tfServQUtil.setEditable(false);

285   GuiAddFields.AddLabel ("Phase:",
286                         false,
287                         0, gridy, 1, 1,
288                         GridBagConstraints.CENTER,
289                         fnt,
290                         gbl, p );
291   tfPhase = GuiAddFields.AddTextField
292       (35,
293        1, gridy, 3, 1,
294        GridBagConstraints.WEST,
295        false,
296        gbl, p );
297   tfPhase.setText("");
298   tfPhase.setEditable(false);

300   tfServQIn  = GuiAddFields.AddTextField
301       (3,
302        4, gridy, 1, 1,
303        GridBagConstraints.WEST,
304        false,
305        gbl, p );
306   tfServQIn.setText("");

```

```

        tfServQIn.setEditable(false);
        tfServQOut = GuiAddFields.AddTextField
            (3,
             5, gridy++, 1, 1,
             GridBagConstraints.WEST,
             false,
             gbl, p );
        tfServQOut.setText("");
        tfServQOut.setEditable(false);
315    GuiAddFields.AddLabel
            ("Pending Read Reqs/Ans",
             false,
             4, gridy++, 2, 1,
             GridBagConstraints.CENTER,
320    fnt,
             gbl, p );
325
//*****GuiAddFields.AddLabel
//*****("Choose Output File:",
//*****false,
//*****0, gridy++, 1, 1,
//*****GridBagConstraints.CENTER,
330    fntBold,
             gbl, p );
            GuiAddFields.AddButton ("Select File...", 0, gridy,
             GridBagConstraints.WEST,
             gbl, p, (Frame)this);
335
tfOutputFile = GuiAddFields.AddTextField
            (45,
             1, gridy, 4, 1,
             GridBagConstraints.WEST,
             false,
             gbl, p );
            tfOutputFile.setEditable(true);
340
            GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
345
//*****GuiAddFields.AddButton ("Trace...", 0, gridy, GridBagConstraints.WEST,
//*****gbl, p, (Frame)this);
            GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
350
//*****GuiAddFields.AddButton ("Save State", 0, gridy,
//*****GridBagConstraints.CENTER,
//*****gbl, p, (Frame)this);
            GuiAddFields.AddButton ("Restore State", 1, gridy,
             GridBagConstraints.CENTER,
             gbl, p, (Frame)this);
355
            searchLearn =
            GuiAddFields.AddButton ("Search & Learn", 3, gridy,
             GridBagConstraints.CENTER,
             gbl, p, (Frame)this);
            searchOnly =
360
            GuiAddFields.AddButton ("Search Only", 5, gridy,
             GridBagConstraints.CENTER,

```

```

                                gbl, p, (Frame)this);

gridy+=2;
370 GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
//*****  

  

375 GuiAddFields.AddButton ("Exit Agent", 2, gridy,
                           GridBagConstraints.CENTER,
                           gbl, p, (Frame)this);

//*****  

  

380 /* general settings */
setBackground(Color.gray);
addWindowListener(new DWAdapter());  

  

scroll.add(p);
add("Center", scroll);
add("West", new Label(" "));
add("East", new Label(" "));
add("South", new Label(" "));
}  

  

390 public void itemStateChanged(ItemEvent event)
{
    if (event.getItemSelectable() == allEngines)
    {
        engines.setVisible(false);
    }
    else if (event.getItemSelectable() == selectEngines)
    {
        engines.setVisible(true);
    }
400 else if (event.getItemSelectable() == textQuery)
    {
        tfQuery.setEditable(true);
        tfRootSetSize.setVisible(true);
        tfQueryRefineFactor.setVisible(true);
        queryRefine.setVisible(true);
        searchLearn.setVisible(true);
        searchOnly.setVisible(true);
        addSites.setVisible(false);
    }
410 else if (event.getItemSelectable() == linkQuery)
    {
        tfQuery.setEditable(true);
        tfRootSetSize.setVisible(false);
        tfQueryRefineFactor.setVisible(false);
        queryRefine.setVisible(false);
        searchLearn.setVisible(true);
        searchOnly.setVisible(true);
        addSites.setVisible(false);
    }
420 else if (event.getItemSelectable() == repSites)
    {
        tfQuery.setEditable(true);
        tfQueryRefineFactor.setVisible(false);
        queryRefine.setVisible(false);
        searchLearn.setVisible(false);
        searchOnly.setVisible(false);
        addSites.setVisible(true);
    }
}

```

```

        }

430

    public void actionPerformed(ActionEvent event)
    {
        String command = event.getActionCommand();

435        if ( command == "Refine Query" )
        {
            Agent.Action action = (Agent.Action) new Agent.ActionRefineQuery
                (tfQuery.getText(),
                 new Integer (tfQueryRefineFactor.getText()).intValue(),
                 tfQuery);
            agent.setAction(action);
        }
        else
445        if ( command == "Add Sites" )
        {
            Agent.Action action = (Agent.Action) new Agent.ActionAddSites
                (tfQuery.getText());
            agent.setAction(action);
        }
        else
450        if (command == "Select File..." )
        {
            FileDialog fd = new FileDialog(this, "Agent File");
            fd.show();
            if (fd.getDirectory() != null && fd.getFile() != null )
                tfOutputFile.setText(fd.getDirectory()+fd.getFile());
        }
        else
460        if (command == "Trace..." )
        {
            TraceDialog td = new TraceDialog(this,
                "Agent Trace Options",
                Trace.AGENT_TRACE,
                trace
                );
            td.show();
        }
        else
470        if (command == "Save State")
        {
            FileDialog fd = new FileDialog(this, "Agent File");
            fd.show();
            if (fd.getDirectory() != null && fd.getFile() != null )
            {
475            String fileName = fd.getDirectory()+fd.getFile();
            try
            {
                ObjectOutputStream oos =
                    new ObjectOutputStream(new FileOutputStream(fileName));
                agent.saveState(oos);
                oos.flush();
                oos.close();
            }
            catch (Exception e)
            {
480                System.err.println("Error while saving the Agent's state:\n"+e);
            }
        }

```

```

        }
490    else
491        if (command == "Restore State")
492        {
493            FileDialog fd = new FileDialog(this, "Agent File");
494            fd.show();
495            if (fd.getDirectory() != null && fd.getFile() != null )
496            {
497                String fileName = fd.getDirectory()+fd.getFile();
498                try
499                {
500                    ObjectInputStream ois =
501                        new ObjectInputStream(new FileInputStream(fileName));
502                    agent.restoreState(ois);
503                    ois.close();
504                }
505                catch (Exception e)
506                {
507                    System.err.println("Error while saving the Agent's state:\n"+e);
508                }
509            }
510        }
511    else
512        if ( command == "Exit Agent")
513        {
514            Agent.Action action = new Agent.ActionExit();
515            agent.setAction(action);
516        }
517    else
518        if (command == "Search & Learn" || command == "Search Only")
519        {
520            boolean update = command == "Search & Learn" ? true : false;
521            boolean goodInput = true;
522            SearchEngine[] searchEngines = null;
523
524            if ( tfOutputFile.getText().length() == 0 )
525            {
526                StatusDialog sd = new StatusDialog
527                    (this,"Bad Input","Output File Must Be Chosen!");
528                sd.show();
529                goodInput = false;
530            }
531            if ( textQuery.getState()           &&
532                 tfRootSetSize.getText().length() == 0 )
533            {
534                StatusDialog sd = new StatusDialog
535                    (this,"Bad Input","Size of Root Set Must Be Specified!");
536                sd.show();
537                goodInput = false;
538            }
539            if ( tfQuery.getText().length() == 0 )
540            {
541                StatusDialog sd = new StatusDialog
542                    (this,"Bad Input","Query Must Be Specified!");
543                sd.show();
544                goodInput = false;
545            }
546            if ( tfExpandFactor.getText().length() == 0 ||
547                 tfNumResults.getText().length() == 0      )
548            {
549                StatusDialog sd = new StatusDialog

```

```

550         (this,"Bad Input","Expand Factor/Num. Results Must Be Set!");
551         sd.show();
552         goodInput = false;
553     }
554     if ( allEngines.getState() )
555     {
556         searchEngines = new SearchEngine [SearchEngine.NUM_ENGINES];
557         try {
558             // prepare an array of all search engines
559             for ( int i = 0; i < SearchEngine.NUM_ENGINES; i++ )
560                 searchEngines[i] = SEManager.ENGINES[i].seClone();
561         }
562         catch (CloneNotSupportedException e)
563             { /* clone is supported by the Search Engine classes*/}
564     }
565     else
566     {
567         int[] selectedEngines = engines.getSelectedIndexes();
568
569         if ( selectedEngines.length == 0 )
570             {
571                 StatusDialog sd = new StatusDialog
572                     (this,"Bad Input","Engines Must Be Chosen!");
573                 sd.show();
574                 goodInput = false;
575             }
576         else
577             {
578                 searchEngines = new SearchEngine[selectedEngines.length];
579                 try
580                 {
581                     // prepare an array with clones of the selected engines
582                     for ( int i=0; i<selectedEngines.length; i++ )
583                         searchEngines[i] = (SearchEngine)
584                             SEManager.ENGINES[selectedEngines[i]].seClone();
585                 }
586                 catch (CloneNotSupportedException e)
587                     { /* clone is supported by the Search Engine classes*/ }
588             }
589         }
590
591         if (goodInput)
592         {
593             Agent.Action action = textQuery.getState() ?
594                 (Agent.Action) new Agent.ActionTextQuery
595                     (tfQuery.getText(),
596                      // Aya 20.10.99 - changed the paradigm, the user must refine
597                      the query
598                          //                                explicitly by clicking the refine query
599                      button
600                          //                                no automatic refinement during query execution
601                          /*
602                           *                                new Integer
603                           (tfQueryRefineFactor.getText()).intValue(), */
604                           0,
605                           new Integer (tfRootSetSize.getText()).intValue(),
606                           new Integer (tfExpandFactor.getText()).intValue(),
607                           new Integer (tfNumResults.getText()).intValue(),
608                           new Float   (tfMaxLinkWeight.getText()).floatValue(),
609                           tfOutputFile.getText(),
610                           searchEngines,
611                           update) :

```

```

        (Agent.Action) new Agent.ActionLinkQuery
            (tfQuery.getText(),
             new Integer (tfExpandFactor.getText()).intValue(),
             new Integer (tfNumResults.getText()).intValue(),
             new Float   (tfMaxLinkWeight.getText()).floatValue(),
             tfOutputFile.getText(),
             searchEngines,
             update);
        agent.setAction(action);
    }
}
else
{
    System.out.println(event);
}

return;
}

630 public final void
setPhase (String phase)
{
    tfPhase.setText(phase);
}
635 public final void
setNSites (int n)
{
    tfNSites.setText(String.valueOf(n));
}
640 public final void
setLeft (int n)
{
    tfLeft.setText(String.valueOf(n));
}
645 public final void
setServQIn (int n)
{
    tfServQIn.setText(String.valueOf(n));
}
650 public final void
setServQOut (int n)
{
    tfServQOut.setText(String.valueOf(n));
}
655 public final void
setServQUtil (int n)
{
    tfServQUtil.setText(String.valueOf(n));
}
660 class DWAdapter extends WindowAdapter
{
    public void windowClosing(WindowEvent event)
    {
        agent.exit();
    }
}
}

```